STREAMIT
IP Media Devices

Lukas Video technical manual

Project name:	IP video
Version:	v1.5
Date:	June, 2015

Document history

Version	Date	Description
1.0	September 2011	Initial version
1.1	December 2011	• Added API descriptions for setting the homepage URL and for multicast streams. • Examples have been updated and corrected.
1.2	December 2011	• Supported stream- and content types. • Firmware update procedure. • Example URL for unicast HTTP Live Stream • Various minor points, incl. unused error codes
1.3	April 2012	• Added description of configuration using a file • Added description of channel list as simple on-board user interface • Added description of custom password feature to protect the start page URL • Added API call that saves the volume setting • Added example to (temporarily) overlay grapics and texts on the video
1.4	August 2012	Updated supported streaming protocols and codecs
1.5	June 2015	Updated references to Wowza product and webpages

Contents

1 Introduction

This document contains the Streamit technical manual for the Lukas Video set-top box. Among other things, this document describes how the Lukas Video can be upgraded (section 3), configured (section 4), how a simple channel list can be used as a user interface (section 5.1) and how you can create your own user interface using the built-in web browser (section 5.2).

Instructions on how to install and configure the Lukas Video box can be found in the Lukas Video User Manual.

2 Supported stream- and content types

The following stream- and contents types are supported by the Lukas Video:

Stream types:

- HTTP Live Streaming
- UDP streaming
- Unicast, multicast and peer to peer streaming

Audio- and video codecs:

- AAC/HE-AAC
- H264

The Lukas Video works very well in combination with Wowza Streaming Engine configured for HTTP Live Streaming. They have clear tutorials on how to configure Wowza Streaming Engine (see `http://www.wowza.com/forums/content.php?4-tutorials`) and how to construct the URL that would also be used for the Lukas Video (see `http://www.wowza.com/forums/content.php?569`). In addition Wowza provides good support on their online forum.

Note that the Lukas Video cannot play a HTTP Live Stream that only contains AAC audio.

3 Firmware upgrade

To use the latest possibilities of the Lukas Video, a firmware update may be required. The latest firmware can be found at the Streamit secure website: `https://secure.streamit.eu`. Send an email to `support@streamit.eu` to request access.

After downloading the latest firmware, take the following steps to upgrade the firmware of the Lukas Video:

- Place update files <version>.dat and <version>.img in the root of a USB stick (make sure there are no other files on the USB stick)
- Power down the Lukas Video and remove the power cord
- Insert the USB stick into the USB connection on the back of the Lukas Video
- Insert the power cord into the Lukas Video and wait untill the 'Update success' message appears
- Remove the power cord
- Remove the USB stick
- Re-insert the power cord and the Lukas Video will boot with the newly installed firmware

4 Configuration

The Lukas Video can be configured in two ways: A file can be loaded from a USB stick, or JavaScript can be loaded from a web page. Both methods are explained in this section.

4.1 Configuration file

The Lukas Video can be configured using a configuration file on a USB stick. There should be only 1 configuration file on the USB stick. To update the Lukas Video, simply insert the USB stick with the configuration file into the Lukas Video. Lukas Video will report success ('Update success') or failure ('Incorrect configuration file') for the configuration update.

The settings in the configuration file on the USB stick will replace the settings on the Lukas Video. This is also the case with the channel list (see section 5.1). All previously configured channels are removed and the channels specified in the configuration file are applied.

4.1.1 File format

The following applies for the format of the configuration file:

- The file is stored in plain text
- The file name must be `lukas_video.scf`
- The file is placed in the root of the USB stick
- Settings are listed one setting per line
- Settings are specified in the form <setting name>=<setting value> (no spaces allowed around '`=`')
- The order in which the settings appear is not important, except for `ch<x>name` and `ch<x>url` that must be specified in that order
- The configuration file must end with an empty line
- Empty lines before the end of the file are ignored
- Lines that contain text that is not a well-formed setting for a known setting name are ignored[1]
- Settings that are not required, are ignored
- Setting names are not case sensitive, so for example `SomeSettingName` is the same as `somesettingname`[2]

[1] This can be used to insert comments in the configuration file. It is recommended to start a comment line with #.
[2] In this document the settings are written in camel case, like 'CamelCase', for readability.

4.1.2 Supported settings

The following settings are supported in the configuration file:

Name	*Description*	*Comment*
Mode	Determines if the Lukas Video connects to a web page (start page) or uses the on-board channel list. Possible values are `StartPage` or `ChannelList`.	This setting is always required.
StartPageUrl	Sets the web page to which the Lukas Video connects at boot up.	This setting is required when the `Mode` is set to `StartPage`.
Ch<*i*>Name	Sets the name for the channel that is accessible as preset <*i*>.	This setting is required when a setting 'Ch<*i*>Url' is present. The name may be empty (`Ch1Name=`). See section 5.1 for details.
Ch<*i*>Url	Sets the URL for the channel that is accessible as preset <*i*>.	See section 5.1 for details.
Ch<*i*>VT	Sets the video type for a multicast stream	This setting is required for a multicast stream. See section 11.3 for values and section 5.1 for details.
Ch<*i*>VP	Sets the PID of the video stream in a multicast stream	This setting is required for a multicast stream. See section 11.3 for values and section 5.1 for details.
Ch<*i*>AT	Sets the audio type for a multicast stream	This setting is required for a multicast stream. See section 11.3 for values and section 5.1 for details.
Ch<*i*>AP	Sets the PID of the audio stream in a multicast stream	This setting is required for a multicast stream. See section 11.3 for values and section 5.1 for details.
Ch<*i*>PT	Sets the PCR type for a multicast stream	This setting is required for a multicast stream and fixed to 9.
Ch<*i*>PP	Sets which PID is used for A/V synchronization.	This setting is required for a multicast stream and should be set to the PID of the video stream.
CustomPw	Supplies the custom password to authorize the changes caused by this configuration file.	This setting is required when the custom password has been set to protect the configuration of the Lukas Video. See section 4.4 for details.
SetCustomPw	Changes the custom password.	When the custom password is already set, the setting `CustomPw` must be present as well with the current custom password. The custom password must be 6 digits. See section 4.4 for details

If you are sure that the configuration file is correct, but the Lukas Video keeps reporting 'Incorrect configuration file', check if you have included the custom password. If the custom password is set, it is required in the configuration file.

4.1.3 Example

The following configuration file is (technically) correct:

```
mode=channellist
StartPageUrl=http://av.streamit.eu/settopbox
ch1name=
ch1url=http://stream1.m3u8
ch2Name=Second channel
ch2url=http://stream2.m3u8
ignored line, could be used as comment
setCustompw=011235
custompw=123123
```

However, it is recommended to improve the configuration file as follows:

```
Mode=ChannelList

Ch1Name=First Channel
Ch1Url=http://stream1.m3u8

Ch2Name=Second channel
Ch2Url=http://stream2.m3u8

CustomPw=123123
SetCustomPw=011235
```

4.2 Web page using the plugin API

4.3 Protection of the start page URL from editing by the end-customer

In some applications of the Lukas Video set-top box it is desirable that the end-customer cannot change the URL of the start page that the Lukas Video loads when starting up. For this a custom password can be set. When a custom password is set, the menu item 'Content source' (showing the start page URL or the channel list) is hidden and cannot be edited through the menu anymore.

The custom password is additional to the standard password. Both passwords are valid, but have a different effect when a factory reset is performed:

	Using standard password	**Using custom password**
AV settings	Set to factory defaults	Set to factory defaults
Network settings	Set to factory defaults	Set to factory defaults
Content source	Not changed	Set to factory defaults
Custom password	Not changed	Not changed

For details on setting a custom password, see section 4.4.

4.4 Setting a custom password

The custom password can be set through the configuration file (see section 4.1), or through a web page that uses the Lukas Video plugin API (see section 4.2).

4.4.1 Through a configuration file

The setting `CustomPw` is required in the configuration file when:

- The configuration file sets the custom password for the first time
- The custom password was already set before. If the custom password has been set before and it is not in the configuration file, no changes are made to the configuration of the Lukas Video.
- The configuration file changes the custom password

To set the custom password the first time, the configuration file must contain the following (next to other settings):

```
CustomPw=123123
SetCustomPw=011235
```

To change the custom password when it currently is `011235`, the configuration file must contain the following (next to other settings):

```
CustomPw=011235
SetCustomPw=135790
```

So: To change the custom password, the 'old' / current custom password must be supplied.

To clear the custom password, it is left empty:

```
CustomPw=011235
SetCustomPw=
```

4.4.2 Through a web page using the plugin API

To set the custom password the first time, the JavaScript call `SetCustomPassword` should be used (see section 7.24). This call is required only once, it does not need to be present in all pages that are served to the Lukas Video.

When the custom password is set, changing the start page URL through the plugin API requires that the JavaScript call `SetStartPageUrl_pwd` is used (see section 7.27).

To change the custom password when it currently is `011235`, two JavaScript calls are needed: First the custom password is cleared using `ClearCustomPassword` (see section 7.1), then the new custom password is set using `SetCustomPassword` (see section 7.24).

To clear the custom password, the JavaScript call `ClearCustomPassword` is used (see section 7.1).

5 User interface

The Lukas Video supports two types of user interface for the end-customer. A simple on-board channel list can be configured, presenting a fixed list of streams by their name (see section 5.1). To create a more sophisticated user interface, the built-in web browser can be used. The Lukas Video then renders the web pages it gets served, allowing for interaction through the remote control (see section 5.2).

5.1 On-board channel list

For a setup where a fixed list of streams is available from which the end-customer can choose, a simple channel list can be configured (see figure ??). This channel list is stored on the Lukas Video itself and does not require any other network access than access to the streams. The maximum number of channels in the list is 10.

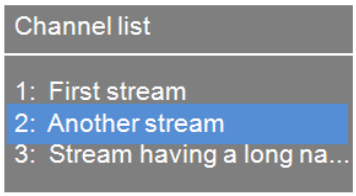

Fig. 1: Example of how the channel list looks on the TV screen

A channel list can be set up using a configuration file (see section 4.1). The following must be observed:

- The settings 'Ch<*i*>Name' and 'Ch<*i*>Url' should always be used together, there should be no 'Ch<*i*>Name' without a 'Ch<*i*>Url' and vice versa.
- When the channel URL points to a multicast stream, the following entries must also be set in the configuration file (see section 4.1.2 for details):
  - Ch<*i*>VT
  - Ch<*i*>VP
  - Ch<*i*>AT
  - Ch<*i*>AP
  - Ch<*i*>PT
  - Ch<*i*>PP
- All previously configured channels are removed and the channels that are specified in the configuration file are applied.
- It is acceptable if no channel is specified, the channel list will be cleared and remain empty.
- The numbering of the channels must be consecutive, for example Ch1Name, Ch2Name, Ch3Name. The following specification is *not* correct:

```
Ch1Name=channel 1
Ch1Url=http://stream1.m3u8
Ch3Name=channel 3
Ch3Url=http://stream3.m3u8
```

- The channels are available as presets via the remote control, so when a user presses '2', the Lukas Video will start playing the URL specified for channel 2.
- If a user presses a number and that channel is not in the list, a red cross is shown in the top left corner of the screen.

5.2 Web-based user interface

The Streamit Lukas Video uses the Qt-WebKit engine to render web pages on television screens. As such, it contains a full set of HTML and JavaScript functionality that any web developer would be familiar with. On top of this, a plugin is available which provides additional functionality that is specific to this device. That functionality can be accessed through JavaScript, and the API is described in detail in section 7.

While web browsers are usually controlled using mouse or touch screen, the Lukas Video uses a remote control. Section 9 explains how web pages can respond to this remote control.

When developing web pages for the Lukas Video, the default URL should be changed to your local domain. Any framework can be used to develop these web pages.

Familiarity with HTML, JavaScript and CSS are assumed throughout this document. Code snippets indicate what language it is in, and it is up to the developer on how to place these either inside an HTML document or into separate files.

Web pages, as served to the Lukas Video, need to be 720 pixels wide and 480 pixels high. The box will scale these pages to the appropriate size, depending on the resolution of the TV screen and the video settings of the box. These settings can be changed in the menu, see the User Manual for details.

The following examples are included in section 10:

- A full example of a basic use case
- A full example of playing a multicast stream
- A skeleton example that (temporarily) overlays grapics and texts on the video

5.3 Serving multiple Lukas Video devices

Setting up a server that can support many Lukas Video set-top boxes requires a web application. Such an app would need to be able to identify a set-top box for authentication purposes. This makes it possible, for instance, to disclose content to some clients, and not to others, or to deny access to set-top boxes you did not sell.

For such purposes, the Lukas Video exposes its MAC address to web applications in two ways: the MAC address is included in the user agent string (which is easy to fake by other devices or browsers) and also the plugin contains a JavaScript function that returns the MAC address of the device (which is much harder to fake, since the user agent would need to have the plugin). See the `GetMac()` function in the JavaScript API on page 12 for details.

When the web application's database contains all MAC addresses of the devices issued by your organization, this mechanism can be used for access control, and to configure each individual device. Optionally, MAC addresses can be coupled to user accounts as well, which would let authorized users manage content access for one or more devices.

6 Using the plugin

The plugin extends the built-in web browser with a configurable video player, audio volume control, device identification and device configuration. This section explains how to load that plugin into web pages, and expose its functionality. Also note that the background of the web page needs to be made transparent in order for the video to be visible.

6.1 Loading the plugin

The plugin can be loaded into a web page by including the following HTML code in the page body:

HTML
```
<object id="pluginobj" type="application/smit-player-plugin"><object>
```

The plugin can then be accessed using JavaScript as follows:

JavaScript
```
var plugin = document.getElementById("pluginobj");
```

Once this variable has been set, it can be used to call the functions which it exposes:

JavaScript
```
var plugin = document.getElementById("pluginobj");
plugin.Fullscreen();
```

6.2 Setting a transparent background

In order for the video player to be visible, the background of the HTML document needs to be transparent. This can be accomplished using CSS:

CSS

```
1  body { background: transparent; }
```

All HTML elements will be rendered on top of the video window, and this is true for the body as well. Setting a non-transparent background will therefore obscure the video.

7 JavaScript API

This section describes each of the functions available in the JavaScript API of the plugin. Note that these functions need to be called on the plugin object, as described in the previous section.

7.1 ClearCustomPassword

Description:
Clear the custom password

Prototype:
`ClearCustomPassword( string password )`

Parameters:

`password`: The current custom password

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
Clears the custom password. See section 4.4 for details.

7.2 FullScreen

Description:
Sets the video display to full screen

Prototype:
`FullScreen()`

7.3 GetBandwidth

Description:
Returns the bandwidth of the current stream

Prototype:
`number GetBandwidth()`

Return value:
The bandwidth

7.4 GetDownLoadSpeed

Description:
Returns the current download speed

Prototype:
`number GetDownLoadSpeed()`

Return value:
The current download speed in kB/s

7.5 GetEndPosition

Description:
Returns the duration of the current media stream

Prototype:
`number GetEndPosition()`

Return value:
The total number of seconds of the stream

Comments:
This function only applies to VOD streams

7.6 GetMac

Description:
Returns the MAC address of the device

Prototype:
`string GetMac()`

Return value:
The MAC address

Comments:
A Media Access Control (MAC) address is a unique identifier for network devices. Note that the user agent string of the Lukas Video browser also contains this MAC address.

7.7 GetMute

Description:
Returns the mute state of the device

Prototype:
`number GetMute()`

Return value:
The mute status: 0 when audio is not muted, 1 when muted

7.8 GetPosition

Description:
Returns the time position of the current media stream

Prototype:
`number GetPosition()`

Return value:
The number of seconds since the start of the stream

Comments:
This function only applies to VOD streams

7.9 GetVideoHeight

Description:
Returns the height of the video window

Prototype:
`number GetVideoHeight()`

Return value:
The current height of the video window

7.10 GetVideoWidth

Description:
Returns the width of the video window

Prototype:
`number GetVideoWidth()`

Return value:
The current width of the video window

7.11 GetVolume

Description:
Returns the current audio volume level

Prototype:
`number GetVolume()`

Return value:
The current volume (ranging between 0 and 100)

7.12 MulticastConnect

Introduced:
firmware version 1.0.10

Description:
Connects to a multicast stream

Prototype:
`number MulticastConnect( string url )`

Parameters:

`url:` The URL of the stream

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
Only one multicast stream can be connected to at any given time. Use MulticastDisconnect() to disconnect from the stream.

7.13 MulticastDisconnect

Introduced:
firmware version 1.0.10

Description:
Disconnects from a multicast stream

Prototype:
`number MulticastDisconnect()`

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
If a multicast stream is playing, call MulticastPlayStop() before disconnecting.

7.14 MulticastPlayStart

Introduced:
firmware version 1.0.10

Description:
Plays the connected multicast stream

Prototype:
`number MulticastPlayStart( number pid, number videoType, number videoPid, number audioType, number audioPid, number pcrType, number pcrPid )`

Parameters:

`pid`: A/V pid; should have value 3 for the Lukas Video

`videoType`: Video type, see table 3 on page 27

`videoPid`: Video pid

`audioType`: Audio type, see table 3 on page 27

`audioPid`: Audio pid

`pcrType`: PCR type, should have value 9 for the Lukas Video

`pcrPid`: PCR pid

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
Use MulticastConnect() to connect to a stream, before using MulticastPlayStart().

7.15 MulticastPlayStop

Introduced:
firmware version 1.0.10

Description:
Stops playing the multicast stream

Prototype:
`number MulticastPlayStop()`

Return value:
Error code. See table 2 on page 26 for an overview

7.16 Pause

Description:
Pauses the current media stream, and freezes the video on the screen

Prototype:
`number Pause()`

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
See the `Resume` command on how to restart the stream

7.17 Play

Description:
Plays a media stream

Prototype:
`number Play( string url )`

Parameters:

`url:` The URL for the stream

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
Supported stream protocol is Apple's HTTP Live Streaming (H264/AAC). It supports both live broadcasts and prerecorded content (video on-demand)

7.18 RefreshVideoWindow

Description:
Refreshes the video window

Prototype:
`number Refresh()`

Return value:
Error code. See table 2 on page 26 for an overview.

Comments:
Call this function after changing the type of video, and after changing the properties of the video.

7.19 RegisterCallbackFun

Description:
Sets an event handler

Prototype:
`number RegisterCallbackFun( string type, object function )`

Parameters:

`type:` Callback type ID. This should be `"player_event"`

`function:` JavaScript function to be called when an event occurs

Return value:
Error code. See table 2 on page 26 for an overview.

Comments:
See section 8 on page 8 on how to catch and process events.

7.20 Resume

Description:
Restarts the media stream after it was paused

Prototype:
`number Restart()`

Return value:
Error code

Comments:
See the `Pause` command on how to pause the stream

7.21 SaveVolume

Description:
Stores the sound volume level to preserve this setting across restarts

Prototype:
`SaveVolume( number level )`

Parameters:

`level:` The volume level: minimum value is 0 (no sound), and maximum value is 84

Return value:
Error code. See table 2 on page 26 for an overview

7.22 Seek

Description:
Sets the time position of the current media stream

Prototype:
`number Seek( number sec )`

Parameters:

`sec:` Number of seconds, relative to the start of the stream

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
This function only applies to VOD streams

7.23 SetBandwidth

Description:
Selects the bandwidth

Prototype:
`number SetBandwidth( number value )`

Parameters:

`value:` Bandwidth in bps

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
This function can be used when the video server playlist offers a stream in multiple bandwidths. If this function is not used, the stream with the lowest bandwidth will be used.

7.24 SetCustomPassword

Description:
Set a custom password

Prototype:
`SetCustomPassword( string password )`

Parameters:

`password:` The desired custom password

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
A custom password protects the factory reset from changing the start page URL and prevents that the start page URL is changed through the menu. See section 4.4 for details.

7.25 SetMute

Description:
Mutes or unmutes the sound

Prototype:
`SetMute( number status )`

Parameters:

`status:` Mute state: 1 to mute, 0 to unmute

Return value:
Error code. See table 2 on page 26 for an overview

Comments:
When unmuting, the volume is restored to the level prior to muting.

7.26 SetStartPageURL

Introduced:
firmware version 1.0.10; changed in version 1.0.24

Description:
Sets the start page URL

Prototype:
`SetStartPageURL( string url )`

Parameters:

`url:` URL of the start page

Return value:
Error code. See table 2 on page 26 for an overview

7.27 SetStartPageURL_pwd

Introduced:
firmware version 1.0.24

Description:
Sets the start page URL when this function has been protected with a custom password. See section 4.4 for details.

Prototype:
`SetStartPageURL_pwd( string url, string password )`

Parameters:

`url`: URL of the start page

`password`: The custom password that was set

Return value:
Error code. See table 2 on page 26 for an overview

7.28 SetVideoHeight

Description:
Sets the height of the video window

Prototype:
`SetVideoHeight ( number height )`

Parameters:

`height`: the height of the video window

Return value:
Error code. See table 2 on page 26 for an overview

7.29 SetVideoPosition

Description:
Sets the position of the video window

Prototype:
`number SetVideoPosition( number x, number y )`

Parameters:

`x`: The x-coordinate

`y`: The y-coordinate

Return value:
Error code. See table 2 on page 26 for an overview

7.30 SetVideoWidth

Description:
Sets the width of the video window

Prototype:
`number SetVideoWidth( number width )`

Parameters:

`width:` the width of the video window

Return value:
Error code. See table 2 on page 26 for an overview

7.31 SetVolume

Description:
Sets the sound volume level

Prototype:
`number SetVolume( number level )`

Parameters:

`level:` The volume level: minimum value is 0 (no sound), and maximum value is 84

Return value:
Error code. See table 2 on page 26 for an overview

7.32 Stop

Description:
Stops playing the current media stream, and removes the video from the screen

Prototype:
`number Stop()`

Return value:
Error code. See table 2 on page 26 for an overview

8 Plugin events

Whenever an event occurs within the plugin, the event handler is called if one has been registered. Registering such a callback function is done as follows:

JavaScript

```
function handleEvent( event_code, return_code ) {
  // code to handle the event...
}

// Obtain the plugin object
var plugin = document.getElementById("pluginobj");

// Register the event handler
plugin.RegisterCallbackFun( "player_event", handleEvent );
```

The event handler takes two parameters. The first one is used to pass the event code; the implemented events and their event codes are summarized in table 1. The second parameter is currently not used.

9 Remote control

Processing key presses from the remote control can be handled the same way as those on a regular keyboard. Using JavaScript, a function can be registered to detect keystrokes:

```
document.onkeydown = handler;
```

where `handler` is the name of the function that processes the event, and takes that event as it's only parameter. Below is an example of key handling, which calls another function if one of the numeric keys is pressed, and ignores all others.

JavaScript

```
1  document.onkeypress = processKeystroke;
2
3  function processKeystroke(event) {
4    var keyCode = event.keyCode;
5    if ( keyCode >= 48 && keyCode <= 57 ) {
6          numericKeyFunction(keyCode-48);
7    }
8  }
```

Table 4 on page 28 contains the full set of key codes for the Streamit Lukas Video remote control.

10 Example code

10.1 Basic example

This section contains a basic, single-page example of how to play a video full screen, using the Lukas Video. The video can be paused and resumed using the remote control (pressing OK is used here to resume).

NOTE: in the `demo.js` file shown below, you will need to insert a valid stream URL in order for this example to work.

HTML

```
<html>
  <head>
    <title>Lukas Video demo</title>
    <style type="text/css">
      body { background: transparent; }
    </style>
    <script type="text/javascript" src="demo.js"></script>
  </head>

  <body>
    <object id="pluginobj" type="application/smit-player-plugin"></object>
  </body>

</html>
```

demo.js

```
// Key codes
const OK    =  13;
const PAUSE = 115;

// Stream URL, e.g., http://<Wowza server>:1935/app/myStream/playlist.m3u8
var stream_url = "<insert URL here>";

// Global variable for the plugin object
var plugin;

function init() {
  // Obtain the plugin object
  plugin = document.getElementById("pluginobj");

  plugin.FullScreen();
  plugin.Play(stream_url);
}

function processKeystroke(event) {
  var keyCode = event.keyCode;
  if ( keyCode == PAUSE ) {
    plugin.Pause();
  }
  else if ( keyCode == OK ) {
    plugin.Resume();
  }
}

// Initialize the player after the page is fully loaded
window.onload = init;

// Register key handler
document.onkeydown = processKeystroke;

```

10.2 Multicast example

The example below plays a multicast stream. Pressing 1 on the remote starts the streams, pressing 2 stops it.

NOTE: You will need to insert a valid stream URL in order for this example to work.

multicast.html

```
<html>
  <head>
    <style type="text/css">
      body { background: transparent; }
    </style>
  </head>

  <body onkeypress="keyHandle()">

    <object id="pluginobj" type="application/smit-player-plugin"></object>

    <script type="text/javascript">

      var player;

      // e.g.: var url = "udp://239.1.1.2:1234";
      var url = "<insert URL here>";

              function keyHandle() {

                if (event.keyCode==49){ // 1 key pressed
                  startPlaying();
                }

                if(event.keyCode==50){ // 2 key pressed
                  player.MulticastPlayStop();
                  player.MulticastDisconnect();
                }

              }

              function startPlaying() {
                player.MulticastConnect(url);
                player.MulticastPlayStart(3, 11, 69, 14, 68, 9, 69);
              }

              window.onload = function() {
                player = document.getElementById("pluginobj");
                player.FullScreen();
                startPlaying();
              }

        </script>

  </body>
</html>
```

10.3 Overlay graphics example

The example below shows how user input can be (temporarily) overlaid on the video.

overlay.html

```
<html>
  <head>
    <style type="text/css">
      body { background: transparent; width: 720px; height: 480px; }
    </style>

    <script type="text/javascript">
      // Key codes
      const K_OK    = 13;
      const K_Pause = 115;

      // Global variable for the timer
      var timer;

      // Register key handler
      document.onkeydown = processKeystroke;

      function processKeystroke( event )
      {
         switch( event.keyCode )
         {
           case K_OK:
             displayKeyPress( "OK" );
             break;
           case K_Pause:
             displayKeyPress( "Pause" );
             break;
           default:
             displayKeyPress( " Some other key " );
         }
       }

       function displayKeyPress( key )
       {
         clearTimeout(timer);

         document.getElementById("keyinfo").style.visibility="visible";
         document.getElementById("keyinfo").firstChild.nodeValue = key;

         timer = setTimeout(
           'document.getElementById("keyinfo").style.visibility="hidden"',3000)
       }
     </script>
   </head>

   <body>
     <div id="keyinfo" style="color: #FFF; margin-top: 50px; margin-left: 50px;">
      </div> <!-- Do not remove the   or the DIV will never be shown -->
   </body>
 </html>
```

11 Tables

All tables are gathered here for easy reference.

11.1 Event codes

Event code	Meaning
0	Unknown event
1	Network error
2 (currently not used)	Server error
3 (currently not used)	Server is unavailable
4 (currently not used)	URL is unavailable
5 (currently not used)	-
6 (currently not used)	Waiting for a connection
7	Connection timeout (e.g., stream does not exist)
8	Buffering
9	Buffer is ok
10	End of VOD stream

Tabel 1: Plugin events

11.2 Error codes

Error code	Meaning
0	No error
1	Repeated initialization
2	Not initialized
3	Parameter error
4	Out of memory
5	Stream is already playing
6	Stream has stopped
7	Invalid operation
8	Unknown error

Tabel 2: Error codes

11.3 Content type codes

Code	Media type	Content type
1	Video	MPEG1
2	Video	MPEG2
3	Video	MPEG4-H264
4	Audio	MPEG 1 Layer I
5	Audio	MPEG 1 Layer II
6	Audio	like HEAAC, Decoder LOAS / LATM - AAC
9	PCR	PCR type
10	Audio	AC3
11	Video	H264
12	Video	MPEG4 Part II
13	Video	Decode Simple/Main/Advanced profile
14	Audio	Decode ADTS - AAC
15	Audio	Decoder LOAS / LATM - AAC
16	Audio	WMA,WMAPRO
17	Audio	DD+ Dolby digital
18	Audio	DTS
19	Video	Multimedia content
20	Audio	Multimedia content
21	Video	AVS Video format
22	AudioDes	MPEG 1 Layer I
23	AudioDes	MPEG 1 Layer II
24	AudioDes	AC3
25	AudioDes	HEAAC
26	Audio	LPCM
27	AuxAudio	MPEG 1 Layer I
28	AuxAudio	MPEG 1 Layer II
29	AuxAudio	like HEAAC,Decoder LOAS / LATM - AAC
30	AuxAudio	AC3
31	AuxAudio	Decode ADTS - AAC
32	AuxAudio	Decoder LOAS / LATM - AAC
33	AuxAudio	WMA,WMAPRO
34	AuxAudio	DD+ Dolby digital
35	AuxAudio	DTS
36	Audio	Dolby PULSE
37	AudioDes	e-AC3
38	AudioDes	Dolby Pulse
39	AuxAudio	Dolby Pulse
40	AuxAudio	LPCM
41	Audio	Independent Sub-stream ID
255	Misc	Non identified pid

Tabel 3: Content type codes

11.4 Key codes for the remote control

Button	Key code
OK	13
Exit	27
Left arrow	37
Up arrow	38
Right arrow	39
Down arrow	40
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
1+	112
Info	113
Pause	115
Volume up	116
Volume down	117
Channel up	118
Channel down	119
Mute	120
Power	121

Tabel 4: Key codes for the remote control